13 Mar 2012

Rubber Duck Problem Solving

At Stack Exchange, **we insist that people who ask questions *put some effort into their question*, and we're kind of jerks about it**. That is, when you set out to ask a question, you should …

- Describe what's happening in sufficient detail that we can follow along. Provide the necessary background for us to understand what's going on, even if we aren't experts in your particular area.
- Tell us why you *need* to know the answer. What led you here? Is it idle curiosity or is this somehow blocking you on a project? We don't need your whole life story, just give us some context here.
- Share your research on your problem; what have you found so far? Why didn't it work? And if you didn't do any research … should you even be asking? If you're inviting us to spend our valuable time helping you, it's only fair that you put in a reasonable amount of your valuable time into crafting a decent question. Help us help you!

We have a great How to Ask page that explains all of

RESOURCES

this, which is linked generously throughout the network. (And on Stack Overflow, due to massive question volume, we actually *force* new users to click through that page before asking their first question. You can see this yourself by asking a question as a new user.)

What we're trying to prevent, most of all, is the unanswerable drive-by question. Those help nobody, and left unchecked they can ruin a Q&A site, turning it into a virtual ghost town. On Stack Exchange, questions that are so devoid of information and context that they can't reasonably be answered will be actively closed, and if they aren't improved, eventually deleted.

Like I said, we're kinda jerks about this. But for good reason: we're not-so-subtly trying to **help you help yourself**, by teaching you Rubber Duck problem solving. And boy does it ever work. I've gotten tons of feedback over the years about how people, in the process of writing up their *thorough, detailed* question for Stack Overflow or another Stack Exchange site, figured out the answer to their own problem.

It's quite common. See for yourself:

> How can I thank the community when I solve my own problems?
>
> I've only posted one question so far, and almost posted another. In both cases, I answered my own questions at least partially while writing it out. I credit the community and the process itself for making me think about the answer. There's nothing explicit in what I'm writing that states quite obviously the answer I needed, but something about writing it down makes me think along extra lines of thought.
>
> Why is it that properly formulating your question often yields you your answer?
>
> I don't know how many times this has happened:
>
> - I have a problem
> - I decide to bring it to stack overflow
> - I awkwardly write down my question

- I realize that the question doesn't make any sense
- I take 15 minutes to rethink how to ask my question
- I realize that I'm attacking the problem from a wrong direction entirely.
- I start from scratch and find my solution quickly.

Does this happen to you? Sometimes asking the right question seems like half the problem.

Beginning to ask a question actually helps me debug my problem myself

Beginning to ask a question actually helps me debug my problem myself, especially while trying to formulate a coherent and detailed enough question body in order to get decent answers. Has this happened to anybody else before?

It's not a new concept, and every community seems to figure it out on their own given enough time, but "Ask the Duck" is a very powerful problem solving technique.

Bob pointed into a corner of the office. "Over there," he said, "is a duck. I want you to ask that duck your question."

I looked at the duck. It was, in fact, stuffed, and very dead. Even if it had not been dead, it probably would not have been a good source of design information. I looked at Bob. Bob was dead serious. He was also my superior, and I wanted to

keep my job.

I awkwardly went to stand next to the duck and bent my head, as if in prayer, to commune with this duck. "What," Bob demanded, "are you doing?"

"I'm asking my question of the duck," I said.

One of Bob's superintendants was in his office. He was grinning like a bastard around his toothpick. "Andy," he said, "I don't want you to pray to the duck. I want you to *ask the duck your question*."

I licked my lips. "Out loud?" I said.

"Out loud," Bob said firmly.

I cleared my throat. "Duck," I began.

"Its name is Bob Junior," Bob's superintendant supplied. I shot him a dirty look.

"Duck," I continued, "I want to know, when you use a clevis hanger, what keeps the sprinkler pipe from jumping out of the clevis when the head discharges, causing the pipe to..."

In the middle of asking the duck my question, the answer hit me. The clevis hanger is suspended from the structure above by a length of all-thread rod. If the pipe-fitter cuts the all-thread rod such that it butts up against the top of the pipe, it essentially will hold the pipe in the hanger and keep it from bucking.

I turned to look at Bob. Bob was nodding. "You know, don't you," he said.

> "You run the all-thread rod to the top of the pipe," I said.
>
> "That's right," said Bob. "Next time you have a question, I want you to come in here and ask the duck, not me. Ask it out loud. If you still don't know the answer, then you can ask me."
>
> "Okay," I said, and got back to work.

I love this particular story because it makes it crystal clear how **the critical part of rubber duck problem solving is to *totally commit* to asking a thorough, detailed question of this imaginary person or inanimate object**. Yes, even if you end up throwing the question away because you eventually realize that you made some dumb mistake. The effort of walking an imaginary someone through your problem, step by step and in some detail, is what will often lead you to your answer. If you aren't willing to put the effort into fully explaining the problem and how you've attacked it, you can't reap the benefits of thinking deeply about your own problem before you ask others to.

If you don't have a coding buddy (but you totally should), you can leverage the Rubber Duck problem solving technique to figure out problems all by yourself, or with the benefit of the greater Internet community. Even if you don't get the answer you wanted, forcing yourself to fully explain your problem – ideally in writing – will frequently lead to new insights and discoveries.

Written by Jeff Atwood

Indoor enthusiast. Co-founder of Stack Exchange and Discourse. Disclaimer: I have no idea what I'm talking about. Find me here: http://twitter.com/codinghorror

Related posts

- Treating User Myopia
- The Case For Case Insensitivity
- New Programming Jargon
- This Is All Your App Is: a Collection of Tiny Details
- I Was a Teenage Hacker

powered by nrelate

Continue Discussion 41 replies
Mar '12

P

Perdervall

While rubber ducking usually is a great idea, I recently had a pretty odd experience with the concept. The company in question was a really tiny one, with a high rotation of developers - at one time the flock of developers were reduced to just one waiting for more to be hired to replace the old worn-out programmers

Now, it was mandated from above that the development team must use scrum, since apparently that solved everything. Even though this scrum team only had one guy on it. The lone developer protested loudly saying, I can't stand in front of the whiteboard talking to myself

whiteboard talking to myself.

So they got him a rubber duck. A really big rubber duck. An actual rubber duck which was introduced to him by the manager saying
"Now you can hold your daily meetings, meet Dennis, your new scrum partner".

It didn't really catch on...

Mar '12

Nonapeptide

How very fitting then that I awarded specialty rubber ducks to the winners of my 2011 ServerFault Challenge. =)

In the process of asking a question on ServerFault or other online communities, I have often come upon the answer to my question. That's a large part of the reason why I started blogging. As I would troubleshoot issues in the real world, I found that writing out my experience would cause me to see the issue in a simpler way and I'd almost always find the solution. By that point, I would have 90% of a blog post written anyway, so I might as well publish it to the benefit of others.

If my post or forum question *didn't* net an answer, well, it usually refined my thought process and made it a valid contribution to the nets at large. Someone, somewhere will then have a much easier time answering it or perhaps learning from what troubleshooting steps I've already taken.

Mar '12

ChristopherA

The reason most of my questions are < 5 is because by the time I've typed out the question, I've found an answer.

Mar '12

SedatK

The only catch is that when you figure out the answer for your question while writing it, you throw the question away. So the community becomes bereft of your self-found answer which might be helpful to people who cannot come up with that obvious answer themselves.

Mar '12

Alex_Esplin

I can't count the number of times I've answered questions or solved bug by trying to explain it to my wife, who is not an engineer.

Breaking your problem or question down so someone who has no or extremely limited domain knowledge works wonders. And you don't look as crazy as you do talking to a duck.

The added bonus is that my wife now has a passing knowledge of all kinds of software-related topics. 

Mar '12

NathanV

For something more interactive, there is http://code-consultant.appspot.com which is an eliza-bot modified to sound more developer-focused.

It will also search stackoverflow for related questions in an attempt to be more helpful than the normal eliza bot.

Mar '12

Joe_Taber

Interesting that at the end of the article you linked to your "Who's Your Coding Buddy?" article from 2009; the last comment (by Jim Howard) on that post mentions something akin to the Rubber Duck principle.

Mar '12

Andand

Reminds me of some sage advice I got... the first and most important step in problem solving is to understand the problem you are trying to solve. The rest just kind of slides into place more often than not.

Mar '12

MartinP

I've had many instances of this starting from a memorable 'water cooler' moment something like 30 years ago. I've thought about it a little bit and I put it down to something with left-brain/right-brain. The act of verbalising moves the problem from one to the other and puts the necessary different perspective on it.

Mar '12

Mason_Gup

This seems to kind of mirror my own internal problem-solving technique. I too have only actually asked a couple of questions because almost every problem I've run into, I've solved myself well before I did enough research and searching to write a quality question that hasn't already been asked a dozen times.

I might even say that Stack Overflow's gotten big enough that it's tough to participate - you almost have to get into kind of obscure stuff to have a question that's meaningful and hasn't already been asked and answered, and you usually have to be a pretty top-level programmer to answer the questions that do pop up quickly enough. Not that I'm really compaining - it's more valuable as a well-indexed and searchable store of knowledge than as question-and-answer resource.

Mar '12

SteveW

We used to use this technique at my last job a lot. Only we called it the dumb rock routine. Usually we'd walk another developer through the problem we're having and end up finding the answer ourselves half the time. In this case the other developer is the "dumb rock". But we could have just as easily substituted a rubber duck.

Or... nowadays I tend to talk to myself a lot. That works too.

Mar '12

SamuelS

Yeah, that's happened to me before, although Do think that the fact that I'm actually typing the question into StackOverflow with the intent to actually ask the question, helps motivate me to go through that exercise.

Mar '12

FacebookU

When I was still in high school, I found it really useful to tell my parents all about the problems that I was having coding, because I'd have to explain everything about what I'm doing to them and lay out what I actually want to do logically in my head.

Mar '12

Hayden_Muhl

One of my old bosses used to do this to me. He would come into my office, get half way through his question, snap his fingers, turn on his heel and go back to his office. He did this a couple times in close succession, and came back to me and said, "You know. I should just get a rock. Before I come to you with a question, I should just ask the rock first."

When I left that job, I got him a rock as a present.

Mar '12

Peter_Walke

Does StackOverflow track the statistics of abandoned questions? I would love to see how many questions are abandoned after a valiant effort (e.g. >10 minutes before of writing, >100 words written). I've definitely had this happen to me.

Mar '12

Leon_Breedt

This is amusing, this has happened to me *so* many times.

Just a few hours ago, I was writing up an iOS programming question, and the exercise of polishing my query and clarifying what I wanted to do led me to the answer, without having to post anything.

Mar '12

Paulmorriss

I worked one place where they had yellow plastic "men" which were put out when the floor had been mopped. We used to say to people - go ask the yellow man. I didn't realise that technique had a proper name. (If it's on wikipedia it must be proper, right?)

Mar '12

DavidR

It doesn't always work, sometimes it just manages to waste a lot of time of everyone involved rather than the other way around. I had this very same problem with StackOverflow. I had a very complicated problem and yet the question could be stated in 2 sentences + an example method signature. As it turns out, what I

wanted was not possible, but instead of getting that answer I -and of course the 3 or 4 people trying to help me- were dragged into discussing the very complicated part because of this rubber ducky philosophy.

Any interaction on SO is so painful for me these days that it has become an absolute last resort, which in turn means I spend as little time there as humanly possible which in turn means I stopped answering anybody else's questions. Clearly, the site is doing fine and doesn't need me in the slightest, but it cannot be denied that my own experience has shifted dramatically from "this is the greatest programmer forum ever!" to "don't subject yourself to SO unless you have no other option."

Mar '12

Moutasema

Hhhh, So lovely article and i will try it my self to see if its true. I dont have a duck on my desk by i have monkey and will start working with it from now on.

Mar '12

Rich_Ryan

I use a similar method to make sure I truly understand something. I create a 5 minute presentation on the subject intended for a public audience. If you can't explain it in 5 minutes you don't really understand it. Similar to the business concept elevator pitch https://en.wikipedia.org/wiki/Elevator_pitch

Mar '12

Ed_Falk

One more technique: I never post a question until I've written the simplest possible program that demonstrates the problem. Most of the time, the act of doing this shows me the solution

If not, then at least I've provided the StackOverflow community with something concrete and simple that they can work from.

Plus it eliminates most of the "have you considered..." non-answers.

Mar '12

DrakeC

A couple of places I worked we called it "A Second Set of Eyes." We hadn't made the leap to an inanimate object.

And, I'm another one who has started to type up a question or two, only to abandon it. Either by ruling out obvious refining questions, or by producing a simple example of the problem.

Mar '12

Smartiala

Regardless of how detailed is your internal representation/understanding of the problem it always makes sense to just verbalise it - my theory is that it just activates different neural passages in your brain which eventually helps in finding the solution.

Mar '12

PeterD

We always called this "talking to the bear", after a plush teddy bear that allegedly at one time. It became traditional, when asking a co-worker a question, to start with "would you mind being the bear for a minute?"

Mar '12

Www

I wish I'd thought of the funny nomenclature, but I didn't. I did however, express this thought back in 2004:

http://www.mooreds.com/wordpress/archives/193

Dan Moore

Mar '12

ChuckW

@Christopher Allen-Poole, if you've gone to the trouble of typing out the question, self-answering is encouraged on SO, so go ahead and give the community the benefit of your experience.

That exact thing happened to me a couple months ago: http://stackoverflow.com/q/8715738/99640

The question wound up so long and involved that it seemed a shame to throw it away, so I posted it and answered it myself.

Mar '12

Jason_Fritz

I love your article, so please ask Bob the Duck out loud: "Why doesn't Stackoverflow have a cancel button on the Ask Question page?"

Mar '12

MarshallG

I have had a duck by my work computer for the past 7 years. It has been a great help in maintaining an old code base.

Mar '12